Security Assessment Report

Marginfi v2 PR109 Kamino

June 25, 2025

Summary

The Sec3 team (formerly Soteria) was engaged to conduct a thorough security analysis of the Marginfi v2 PR109 Kamino smart contracts.

The artifact of the audit was the source code of the following programs, excluding tests, in a private repository.

The initial audit focused on the following versions and revealed 5 issues or questions.

program	type	commit
Marginfi v2 PR109	Solana	`61dd5fba4ffaf11308feeccbccd6e97f3cdb1dbc`

This report provides a detailed description of the findings and their respective resolutions.

Table of Contents

Result Overview

Issue	Impact	Status
MARGINFI V2 PR109		
[M-01] Obligation initialization fails for reserves with farms	Medium	Resolved
[L-01] Insufficient reward destination token account check	Low	Resolved
[L-02] Account balances left unsorted after Kamino operations	Low	Resolved
[I-01] Missing refresh_obligation before Kamino deposit/withdraw	Info	Acknowledged
[I-02] Lack of referrer restriction in obligation initialization	Info	Resolved

Findings in Detail

MARGINFI V2 PR109

[M-01] Obligation initialization fails for reserves with farms

To ensure an obligation remains active, the `init_obligation` instruction deposits a minimum of 10 units of collateral. However, when preparing the deposit CPI, the current implementation does not provide valid `obligation_farm_user_state` and `reserve_farm_state` accounts.

```
/* programs/marginfi/src/instructions/kamino/init_obligation.rs */
285 | // --- optional “farms_accounts” group ---
286 | let farms_accounts = SocializeLossV2FarmsAccounts {
287 |     obligation_farm_user_state: None, // or .into()
288 |     reserve_farm_state: None,
289 | };
```

Klend's v2 deposit instruction requires these two accounts to be provided for any reserve that has an associated farm, as they are needed for a refresh operation. This requirement means the current implementation of `init_obligation` does not support reserves with active farms.

It is recommended to update the implementation to pass the corresponding accounts on an as-needed basis, referencing the logic within the `kamino_deposit` instruction.

Resolution

This issue has been fixed by `09b2a6f`.

MARGINFI V2 PR109

[L-01] Insufficient reward destination token account check

The `kamino_harvest_reward` instruction allows a caller to claim rewards through a CPI to the `harvest_reward` instruction of the Kamino farm program. The claimed rewards are then transferred to a token account belonging to the global fee admin.

```
/* programs/marginfi/src/instructions/kamino/harvest_reward.rs */
042 | /// Destination token account must be owned by the global fee admin
043 | #[account(
044 |     mut,
045 |     constraint = destination_token_account.owner == fee_state.load()?.global_fee_admin @
↪  MarginfiError::Unauthorized
046 | )]
047 | pub destination_token_account: Box<InterfaceAccount<'info, TokenAccount>>,
```

However, the current implementation does not enforce that this destination token account must be a specific, fixed token account or an ATA. This issue allows a malicious user to create a new token account for the global fee admin and direct the rewards to it. Consequently, this could impede the global fee admin's ability to consolidate all collected rewards.

```
/* programs/marginfi/src/lib.rs */
476 | /// (fee admin only) Harvest the specified reward index from the Kamino Farm attached to this bank.
477 | ///
478 | /// * `reward_index` - index of the reward token in the Kamino Farm's reward list
479 | pub fn kamino_harvest_reward(
```

It is recommended to restrict the `destination_token_account` to an ATA or, as suggested in the code comments, to limit the execution of this instruction exclusively to the global fee admin.

Resolution

This issue has been fixed by `a262057`.

MARGINFI V2 PR109

[L-02] Account balances left unsorted after Kamino operations

Subsequent to the Marginfi v0.1.3, a `sort_balances` operation is executed following any action that structurally alters an account's balances, such as creating a new balance record or closing an existing one. This procedure ensures the account's balances remain gapless and sorted, which is a invariant for the proper functioning of the risk engine and other instructions.

However, this `sort_balances` operation is absent from the `kamino_deposit` and `kamino_withdraw` instructions.

Resolution

This issue has been fixed by `92c008b`.

MARGINFI V2 PR109

[I-01] Missing refresh_obligation before Kamino deposit/withdraw

In the `init_obligation` flow, `refresh_obligation` is invoked before performing a deposit.

```
/* programs/marginfi/src/instructions/kamino/init_obligation.rs */
040 | // Refresh obligation is needed before a deposit can be made
041 | ctx.accounts.cpi_refresh_obligation()?;
042 | // Transfer tokens from user (signer_token_account) -> obligation owner (liquidity vault)
043 | ctx.accounts.cpi_transfer_user_to_obligation_owner(amount)?;
044 | // Deposit into Kamino (liquidity vault) -> (reserve_liquidity_supply)
045 | ctx.accounts.cpi_kamino_deposit(amount)?;
```

However, `cpi_refresh_obligation` is not invoked prior to the primary operations in the `kamino_deposit` and `kamino_withdraw` instructions.

Given that Kamino's implementation of deposit and withdraw requires the associated obligation to be refreshed within the same slot, it is recommended to add a `cpi_refresh_obligation` call at the beginning of the `kamino_deposit` and `kamino_withdraw` instructions.

Resolution

The team clarified that it is intended to left the `refresh_obligation` call as a separate instruction to be called in the same transaction.

MARGINFI V2 PR109

[I-02] Lack of referrer restriction in obligation initialization

Any user can call the `kamino_init_obligation` instruction to perform the necessary Kamino-side initialization after the group admin has executed `add_pool`.

```
/* programs/marginfi/src/instructions/kamino/init_obligation.rs */
115 | /// We may pass in a mfi controlled account as the referrer
116 | /// CHECK: validated by the Kamino program, generally unrestricted.
117 | #[account(mut)]
118 | pub referrer_user_metadata: Option<UncheckedAccount<'info>>,
```

Although several accounts are verified through the bank, and Kamino itself performs certain checks, the `referrer_user_metadata` can still be arbitrarily supplied by the caller. A malicious attacker could preemptively call `kamino_init_obligation` and provide their own information as the `referrer_user_metadata`. While the current implementation only uses Kamino's deposit and withdraw functions which does not generate referrer fees, it is recommended to impose restrictions on this account.

Resolution

This issue has been fixed by `76682ce`.

Appendix: Methodology and Scope of Work

Assisted by the Sec3 Scanner developed in-house, the manual audit particularly focused on the following work items:

- Check common security issues.
- Check program logic implementation against available design specifications.
- Check poor coding practices and unsafe behavior.
- The soundness of the economics design and algorithm is out of scope of this work

DISCLAIMER

The instance report ("Report") was prepared pursuant to an agreement between Coderrect Inc. d/b/a Sec3 (the "Company") and MRGN, Inc. (the "Client''). This Report solely includes the results of a technical assessment of a specific build and/or version of the Client's code specified in the Report ("Assessed Code") by the Company. The sole purpose of the Report is to provide the Client with the results of the technical assessment of the Assessed Code. The Report does not apply to any other version and/or build of the Assessed Code. Regardless of the contents of the Report, the Report does not (and should not be interpreted to) provide any warranty, representation or covenant that the Assessed Code: (i) is error and/or bug free, (ii) has no security vulnerabilities, and/or (iii) does not infringe any third-party rights. Moreover, the Report is not, and should not be considered, an endorsement by the Company of the Assessed Code and/or of the Client. Finally, the Report should not be considered investment advice or a recommendation to invest in the Assessed Code and/or the Client.

This Report is considered null and void if the Report (or any portion thereof) is altered in any manner.

ABOUT

The Sec3 audit team comprises a group of computer science professors, researchers, and industry veterans with extensive experience in smart contract security, program analysis, testing, and formal verification. We are also building automated security tools that incorporate static analysis, penetration testing, and formal verification.

At Sec3, we identify and eliminate security vulnerabilities through the most rigorous process and aided by the most advanced analysis tools.

For more information, check out our website and follow us on twitter.